

# ESRI\_R3.DLL

Version 1.0.0.1

## Introduction

This document describes the functions contained in esri\_r3.dll. This DLL is used by a number of interfaces between ESRI GIS and SAP R/3, including the ESRI Data Query for R/3 and the ArcView GIS Interface to R/3. The DLL allows connections to R/3 via remote function calls (RFCs) or via the business application programming interface (BAPI) of business objects (BOs).

This document contains the following sections:

<b>Contents</b>	<b>Page</b>
Exported Function Declarations	2
Explanation of Input/Output Parameters	6
Return Values for All Functions	12
Using the Functions Together	13
Special Notes and Data Types	14
ArcView GIS Sample Avenue Script	15
Sample VB Module Declarations	18

## Compatibility

The DLL works with Windows 95 and Windows NT 4.0.

The RFC functionality is compatible from R/3 3.X to 4.0b.

The BAPI functionality is compatible from R/3 3.1h to 4.0b.

## Who should read this document?

This document is intended for the application developer (i.e. ArcView Avenue or VB developer) and an R/3 function module and external integration developer. When interfacing ArcView GIS or any other application (i.e. VB projects) with R/3, it is rare to find a single person who knows both the application software and R/3 (architecture and external integration). It is likely that the application developer may need to call for the assistance of an R/3 developer or system administrator.

```

/*****
*      Exported Function Declarations
*****/

/*      Connection functions      */
/*****/

int R3Logon(char* Client, char* User, char* Password, char* Language, char* Destination,
            char* HostName, char* Gateway, char* Service, int SystemNo)

int R3Logoff()

/*      Connection functions for Visual Basic      */
/*****/

int vbR3Logon(ByVal Client, ByVal User, ByVal Password, ByVal Language,
              ByVal Destination, ByVal HostName, ByVal Gateway, ByVal Service, int SystemNo)

int vbR3Logoff()

/*      BAPI functions      */
/*****/

int InitializeBO(char* BOName)

Int SetBAPIKey(char* BOName, char* BAPIName, char* Parameter, char* Value)

int SetBAPIImport(char* BOName, char* BAPIName, char* Parameter, char* Value)

int SetBAPIImportField(char* BOName, char* BAPIName, char* StructureName, char* Field,
                       char* Value)

int AddBAPITableRow(char* BOName, char* BAPIName, char* TableName)

int SetBAPITableRow(char* BOName, char* BAPIName, char* TableName, int RowNumber)

int SetBAPITableField(char* BOName, char* BAPIName, char* TableName, char* ColumnName,
                      char* Value)

int CallBAPI(char* BOName, char* BAPIName)

int GetBAPIExport(char* BOName, char* BAPIName, char* Parameter, char* Result, char* Type)

int GetBAPIExportField(char* BOName, char* BAPIName, char* Structure, char* Field, char* Result,
                       char* Type)

int GetBAPITableRowCount(char* BOName, char* BAPIName, char* TableName, char* ReturnCode)

int GetBAPITableField(char* BOName, char* BAPIName, char* TableName, char* ColumnName,
                      char* Result, char* Type)

int ClearBAPITable(char* BOName, char* BAPIName, char* TableName)

int DumpBAPITableToDBase(char* BOName, char* BAPIName, char* TableName, char* FileName)

```

```

/*      BAPI functions for Visual Basic      */
/*****/

Long vbInitializeBO(ByVal BOName as String)

Long vbSetBAPIKey(ByVal BOName as String, ByVal BAPIName as String, ByVal Parameter as
String, ByVal Value as String)

Long vbSetBAPIImport(ByVal BOName as String, ByVal BAPIName as String, ByVal Parameter
as String, ByVal Value as String)

Long vbSetBAPIImportField(ByVal BOName as String, ByVal BAPIName as String, ByVal
StructureName as String, ByVal Field as String, ByVal Value as String)

Long vbAddBAPITableRow(ByVal BOName as String, ByVal BAPIName as String, ByVal
TableName as String)

Long vbSetBAPITableRow(ByVal BOName as String, ByVal BAPIName as String, ByVal
TableName as String, ByVal RowNumber as Long)

Long vbSetBAPITableField(ByVal BOName as String, ByVal BAPIName as String, ByVal
TableName as String, ByVal ColumnName as String, ByVal Value as String)

Long vbCallBAPI(ByVal BOName as String, ByVal BAPIName as String)

Long vbGetBAPIExport(ByVal BOName as String, ByVal BAPIName as String, ByVal Parameter
as String, ByVal Result as String, ByVal Type as String)

Long vbGetBAPIExportField(ByVal BOName as String, ByVal BAPIName as String, ByVal
Structure as String, ByVal Field as String, ByVal Result as String, ByVal Type as String)

Long vbGetBAPITableRowCount(ByVal BOName as String, ByVal BAPIName as String, ByVal
TableName as String, ByVal ReturnCode as String)

Long vbGetBAPITableField(ByVal BOName as String, ByVal BAPIName as String, ByVal
TableName as String, ByVal ColumnName as String, ByVal Result as String, ByVal Type as
String)

Long vbClearBAPITable(ByVal BOName as String, ByVal BAPIName as String, ByVal
TableName as String);

Long vbDumpBAPITableToDBase(ByVal BOName as String, ByVal BAPIName as String, ByVal
TableName as String, ByVal FileName as String)

```

```

/*      RFC functions      */
/*****/

int InitializeRFC(char* RFCName)

int SetRFCImport(char* RFCName, char* Parameter, char* Value)

int SetRFCImportField(char* RFCName, char* StructureName, char* Field, char* Value)

int AddRFCTableRow(char* RFCName, char* TableName)

int SetRFCTableRow(char* RFCName, char* TableName, int RowNumber)

int SetRFCTableField(char* RFCName, char* TableName, char* ColumnName, char* Value)

int CallRFC(char* RFCName)

int GetRFCEXport(char* RFCName, char* Parameter, char* Result, char* Type)

int GetRFCEXportField(char* RFCName, char* StructureName, char* Field, char* Result, char* Type)

int GetRFCTableRowCount(char* RFCName, char* TableName, char* ReturnCode)

int GetRFCTableField(char* RFCName, char* TableName, char* ColumnName, char* Result, char* Type)

int ClearRFCTable(char* RFCName, char* TableName)

int DumpRFCTableToDBase(char* RFCName, char* TableName, char* FileName)

/*      RFC functions for Visual Basic      */
/*****/

Long vbInitializeRFC(ByVal RFCName as String)

Long vbSetRFCImport(ByVal RFCName as String, ByVal Parameter as String, ByVal Value as
String)

Long vbSetRFCImportField(ByVal RFCName as String, ByVal StructureName as String, ByVal
Field as String, ByVal Value as String)

Long vbAddRFCTableRow(ByVal RFCName as String, ByVal TableName as String)

Long vbSetRFCTableRow(ByVal RFCName as String, ByVal TableName as String, ByVal
RowNumber as Long)

Long vbSetRFCTableField(ByVal RFCName as String, ByVal TableName as String, ByVal
ColumnName as String, ByVal Value as String)

Long vbCallRFC(ByVal RFCName as String)

Long vbGetRFCEXport(ByVal RFCName as String, ByVal Parameter as String, ByVal Result as
String, ByVal Type as String)

Long vbGetRFCEXportField(ByVal RFCName as String, ByVal StructureName as String, ByVal
Field as String, ByVal Result as String, ByVal Type as String)

```

Long vbGetRFCTableRowCount(ByVal RFCName as String, ByVal TableName as String, ByVal  
ReturnCode as String)

Long vbGetRFCTableField(ByVal RFCName as String, ByVal TableName as String, ByVal  
ColumnName as String, ByVal Result as String, ByVal Type as String)

Long vbClearRFCTable(ByVal RFCName as String, ByVal TableName as String)

Long vbDumpRFCTableToDBase(ByVal RFCName as String, ByVal TableName as String, ByVal  
FileName as String)

```

/*****
*      Explanation of input/output parameters
*****/

```

```

/*      Connection functions      */
/*****/

```

#### R3Logon/vbR3Logon

Client: R/3 Client (e.g., "001").

User: R/3 user account.

Password: R/3 password for user account.

Language: R/3 available language (e.g., "E").

Destination: Put a three-character identifier (e.g., "DLL").

HostName: R/3 server name or IP address.

Gateway: R/3 gateway server. If there is no separate gateway, use the R/3 server name or IP address.

Service: The name of the R/3 gateway network service to use (e.g., sapgw03). If there is no separate gateway, derive the service name from the SystemNo (e.g., if SystemNo = "02" then Service = "sapgw02").

SystemNo: The R/3 system number.

#### R3Logoff/vbR3Logoff

There are no parameters for these functions.

```

/*      BAPI functions      */
/*****/

```

#### InitializeBO/vbInitializeBO

BOName: The name of the R/3 Business Object.

#### SetBAPIKey/vbSetBAPIKey

BOName: The name of the R/3 Business Object.

BAPIName: The name of the BAPI within the Business Object.

Parameter: The name of the Key parameter of the Business Object.

Value: The value to be set for the parameter.

**SetBAPIImport/vbSetBAPIImport**

**BOName:** The name of the R/3 Business Object.  
**BAPIName:** The name of the BAPI within the Business Object.  
**Parameter:** The name of the Import parameter of the BAPI.  
**Value:** The value to be set for the parameter.

**SetBAPIImportField/vbSetBAPIImportField**

**BOName:** The name of the R/3 Business Object.  
**BAPIName:** The name of the BAPI within the Business Object.  
**StructureName:** The name of the Import structure parameter of the BAPI.  
**Field:** The field name in the structure.  
**Value:** The value to be set for the field.

**AddBAPITableRow/vbAddBAPITableRow**

**BOName:** The name of the R/3 Business Object.  
**BAPIName:** The name of the BAPI within the Business Object.  
**TableName:** The name of the Table parameter of the BAPI.

*Note: The added row will be appended to the table. The row number of the new row will be the number of rows of the table. Use GetBAPITableRowCount/vbGetBAPITableRowCount function to get this information.*

**SetBAPITableRow/vbSetBAPITableRow**

**BOName:** The name of the R/3 Business Object.  
**BAPIName:** The name of the BAPI within the Business Object.  
**TableName:** The name of the Table parameter of the BAPI.  
**RowNumber:** The row number to set the table row pointer to.

**SetBAPITableField/vbSetBAPITableField**

**BOName:** The name of the R/3 Business Object.  
**BAPIName:** The name of the BAPI within the Business Object.  
**TableName:** The name of the Table parameter of the BAPI.  
**ColumnName:** The name of the column of the field.  
**Value:** The value to be set for the field.

*Note: The row is set using the SetBAPITableRow/vbSetBAPITableRow function.*



**CallBAPI/vbCallBAPI**

BOName: The name of the R/3 Business Object.  
 BAPIName: The name of the BAPI within the Business Object.

**GetBAPIExport/vbGetBAPIExport**

BOName: The name of the R/3 Business Object.  
 BAPIName: The name of the BAPI within the Business Object.  
 Parameter: The name of the Import parameter of the BAPI.  
 Result: If the function was successful, then this parameter will contain the result as a character string.  
 Type: If the function was successful, then this parameter will contain the result type (i.e., CHR for strings, INT for integers, FLT for float numbers).

**GetBAPIExportField/vbGetBAPIExportField**

BOName: The name of the R/3 Business Object.  
 BAPIName: The name of the BAPI within the Business Object.  
 Structure: The name of the Structure parameter of the BAPI.  
 Field: The field name in the structure.  
 Result: If the function was successful, then this parameter will contain the result as a character string,  
 Type: If the function was successful, then this parameter will contain the result type (i.e., CHR for strings, INT for integers, FLT for float numbers).

**GetBAPITableRowCount/vbGetBAPITableRowCount**

BOName: The name of the R/3 Business Object.  
 BAPIName: The name of the BAPI within the Business Object.  
 TableName: The name of the Table parameter of the BAPI.  
 ReturnCode: If the function was successful, then this parameter will contain "OK." Otherwise it will contain "ERROR."

*Note: If the function was successful, then the return value is the result. Otherwise, the return value is an error code.*

**GetBAPITableField/vbGetBAPITableField**

**BOName:** The name of the R/3 Business Object.  
**BAPIName:** The name of the BAPI within the Business Object.  
**TableName:** The name of the Table parameter of the BAPI.  
**ColumnName:** The name of the column of the field.  
**Result:** If the function was successful, then this parameter will contain the result as a character string.  
**Type:** If the function was successful, then this parameter will contain the result type (i.e., CHR for strings, INT for integers, FLT for float numbers).

*Note: The row is set using the SetBAPITableRow/vbSetBAPITableRow function.*

**ClearBAPITable/vbClearBAPITable**

**BOName:** The name of the R/3 Business Object.  
**BAPIName:** The name of the BAPI within the Business Object.  
**TableName:** The name of the Table parameter of the BAPI.

**DumpBAPITableToDBase/vbDumpBAPITableToDBase**

**BOName:** The name of the R/3 Business Object.  
**BAPIName:** The name of the BAPI within the Business Object.  
**TableName:** The name of the Table parameter of the BAPI.  
**FileName:** The path and file name of the dBASE (.dbf) file.

```

/*      RFC functions      */
/*****
  
```

**InitializeRFC/vbInitializeRFC**

**RFCName:** The name of the R/3 Remote Function Module.

**SetRFCImport/vbSetRFCImport**

**RFCName:** The name of the R/3 Remote Function Module.  
**Parameter:** The name of the Import parameter of the RFC.  
**Value:** The value to be set for the parameter.

**SetRFCImportField/vbSetRFCImportField**

**RFCName:** The name of the R/3 Remote Function Module.  
**StructureName:** The name of the Import structure parameter of the RFC.  
**Field:** The field name in the structure.  
**Value:** The value to be set for the field.

## AddRFCTableRow/vbAddRFCTableRow

RFCName: The name of the R/3 Remote Function Module.  
 TableName: The name of the Table parameter of the RFC.

## SetRFCTableRow/vbSetRFCTableRow

RFCName: The name of the R/3 Remote Function Module.  
 TableName: The name of the Table parameter of the RFC.  
 RowNumber: The row number to set the table row pointer to.

## SetRFCTableField/vbSetRFCTableField

RFCName: The name of the R/3 Remote Function Module.  
 TableName: The name of the Table parameter of the RFC.  
 ColumnName: The name of the column of the field.  
 Value: The value to be set for the field.

*Note: The row is set using the SetRFCTableRow/vbSetRFCTableRow function.*

## CallRFC/vbCallRFCI

RFCName: The name of the R/3 Remote Function Module.

## GetRFCExport/vbGetRFCIExport

RFCName: The name of the R/3 Remote Function Module.  
 Parameter: The name of the Import parameter of the RFC.  
 Result: If the function was successful, then this parameter will contain the result as a character string.  
 Type: If the function was successful, then this parameter will contain the result type (i.e., CHR for strings, INT for integers, FLT for float numbers).

## GetRFCExportField/vbGetRFCExportField

RFCName: The name of the R/3 Remote Function Module.  
 Structure: The name of the Structure parameter of the RFC.  
 Field: The field name in the structure.  
 Result: If the function was successful, then this parameter will contain the result as a character string.  
 Type: If the function was successful, then this parameter will contain the result type (i.e., CHR for strings, INT for integers, FLT for float numbers).

#### GetRFCTableRowCount/vbGetRFCTableRowCount

RFCName: The name of the R/3 Remote Function Module.  
 TableName: The name of the Table parameter of the BAPI.  
 ReturnCode: If the function was successful, then this parameter will contain “OK.” Otherwise it will contain “ERROR.”

*Note: If the function was successful, then the return value is the result. Otherwise, the return value is an error code.*

#### GetRFCTableField/vbGetRFCTableField

RFCName: The name of the R/3 Remote Function Module.  
 TableName: The name of the Table parameter of the RFC.  
 ColumnName: The name of the column of the field.  
 Result: If the function was successful, then this parameter will contain the result as a character string.  
 Type: If the function was successful, then this parameter will contain the result type (i.e., CHR for strings, INT for integers, FLT for float numbers).

*Note: The row is set using the SetRFCTableRow/vbSetRFCTableRow function.*

#### ClearRFCTable/vbClearRFCTable

RFCName: The name of the R/3 Remote Function Module.  
 TableName: The name of the Table parameter of the RFC.

#### DumpRFCTableToDBase/vbDumpRFCTableToDBase

RFCName: The name of the R/3 Remote Function Module.  
 TableName: The name of the Table parameter of the RFC.  
 FileName: The path and file name of the dBASE (.dbf) file.

#### DumpSQLTableToDBase/vbDumpSQLTableToDBase

FileName: The path and file name of the dBASE (.dbf) file.

*Note: This function is used to download the “DATA” table of the “RFC\_READ\_TABLE” R/3 function module to a dBASE file.*

```

/*****

```

Explanation of return values for all functions.

```

*****/

```

```

0 // Success
1 // RFC error see Eesri_r3.err file.
2 // R/3 not connected or logon failed.
3 // Problem creating connection structure.
4 // Parameter not found.
5 // Table not found.
6 // Parameter is not a simple parameter.
7 // Parameter is not a structure.
8 // Error creating file.
9 // Free memory failed when deleting function.
10 // Call of function failed.
11 // RFC exception.
12 // Invalid row.
13 // Invalid column.
14 // Field not found.
15 // Empty table.
16 // Malloc failed.
17 // Failed to create a dBASE file.
18 // Failed to access dBASE field.
19 // Failed to add field to dBASE file.
20 // Failed to write record into dBASE file.
21 // Business Object has already been initialized.
22 // Business Object initialization failed.
23 // Business Object collection maxed out.
24 // Business Object not initialized.
25 // BAPI call failed.
26 // Cannot create BAPI Parameter.
27 // RFC has already been initialized.
28 // RFC collection maxed out.
29 // RFC initialization failed.
30 // RFC not initialized.
31 // Problem accessing RFC parameter.
32 // Problem accessing RFC Table.
33 // An R/3 data type is not supported.
34 // The Table has no fields.

```

```

/*****

```

```

*      How to use the functions together?

```

```

*****/

```

Use the functions with “regular” names such as “R3Logon” for applications such as ArcView GIS or C/C++. Use the functions with the “vb” prefix such as “vbR3Logon” for all Visual Basic applications. The following text uses the “regular” names but is applicable to the “vb” prefixed functions.

#### For BAPIs

- Logon to R/3 using R3Logon.
- Initialize the R/3 business object using InitializeBO.
- If applicable, set key fields using SetBAPIKey.
- If applicable, set all required import parameters using SetBAPIImport or SetBAPIImportField.
- If applicable, set all required tables. First append a row to the table using AddBAPITableRow. Then, set the required fields of that row using SetBAPITableRow and SetBAPITableField.
- Then, call the function using CallBAPI.
- If the call is successful, then access results through export parameters using GetBAPIExport or GetBAPIExportField. To access tables, first determine the row count using GetBAPITableRowCount. Then to access a field, set the row pointer to a particular row (1 to row count) using SetBAPITableRow. Finally, access the fields using GetBAPITableField. You can dump the whole table to a dBASE file using DumpBAPITableToDBase.
- After using a BAPI, clear all tables by using ClearBAPITable.

#### For RFCs

- Logon to R/3 using R3Logon.
- Initialize the R/3 business object using InitializeRFC.
- If applicable, set all required import parameters using SetRFCIImport or SetRFCIImportField.
- If applicable, set all required tables. First append a row to the table using AddRFCTableRow. Then, set the required fields of that row using SetRFCTableRow and SetRFCTableField.
- Then, call the function using CallRFC.
- If the call is successful, then access results through export parameters using GetRFCExport or GetRFCExportField. To access tables, first determine the row count using GetRFCTableRowCount. Then to access a field, set the row pointer to a particular row (1 to row count) using SetRFCTableRow. Finally, access the fields using GetRFCTableField. You can dump the whole table to a dBASE file using DumpRFCTableToDBase.
- After using a function module, clear all tables by using ClearRFCTable.

#### Special Notes:

Once a Business Object or an RFC has been initialized and used, it can be used again without having to reinitialize it.

Make sure that all import and table parameters are cleared or set appropriately before reusing the RFC or BAPI.

A maximum of 100 Business Objects and 100 RFCs can be initialized and used (limited by system memory, of course).

#### R/3 Data Types:

R/3 data type TYPP (i.e., FLT<sup>\*</sup>) is supported and converted to strings and vice versa.

R/3 data types TYP, TYPNUM, TYPDATE and TYPTIME (i.e. CHR<sup>\*</sup>) are supported and converted to strings and vice versa.

R/3 data types TYPINT, TYPINT1, TYPINT2 and TYPFX (with length <= 4) (i.e. INT<sup>\*</sup>) are supported and converted to strings and vice versa.

R/3 data type TYPFLOAT (i.e., FLT<sup>\*</sup>) is supported and converted to strings and vice versa.

<sup>\*</sup>CHR, INT, and FLT are possible values of the Type parameter of GetRFCExport, GetRFCExportField, and GetRFCTableField.

**Below is an example of an ArcView GIS Avenue script that declares the RFC functions of the DLL.**

---

```

myDLL = DLL.Make("C:\ESRI\SHARED\ESRI_R3.dll".asFileName)
if (_myDLL = nil) then
    MsgBox.Info("ESRI_R3.DLL initialization failed.", "")
    return nil
end
_R3Logon = DLLProc.Make(_myDLL, "R3Logon", #DLLPROC_TYPE_INT32,
    {#DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR,
    #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR,
    #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR,
    #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR,
    #DLLPROC_TYPE_INT32})
if (_R3Logon = nil) then
    MsgBox.Info("Error: R3Logon", "")
    return nil
end
_R3Logoff = DLLProc.Make(_myDLL, "R3Logoff", #DLLPROC_TYPE_INT32, {})
if (_R3Logoff = nil) then
    MsgBox.Info("Error: R3Logoff", "")
    return nil
end
_InitializeBO = DLLProc.Make(_myDLL, "InitializeBO", #DLLPROC_TYPE_INT32,
    {#DLLPROC_TYPE_STR})
if (_InitializeBO = nil) then
    MsgBox.Info("Error: InitializeBO", "")
    return nil
end
_SetBAPIImport = DLLProc.Make(_myDLL, "SetBAPIImport", #DLLPROC_TYPE_INT32,
    {#DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR,
    #DLLPROC_TYPE_STR})
if (_SetBAPIImport = nil) then
    MsgBox.Info("Error: SetBAPIImport", "")
    return nil
end
_SetBAPIImportField = DLLProc.Make(_myDLL, "SetBAPIImportField", #DLLPROC_TYPE_INT32,

```



```

        {#DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR,
        #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR})

if (_SetBAPIImportField = nil) then
    MsgBox.Info("Error: SetBAPIImportField", "")
    return nil
end
_CallBAPI = DLLProc.Make(_myDLL, "CallBAPI", #DLLPROC_TYPE_INT32,
    {#DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR})
if (_CallBAPI = nil) then
    MsgBox.Info("Error: CallBAPI", "")
    return nil
end
_GetBAPIExport = DLLProc.Make(_myDLL, "GetBAPIExport", #DLLPROC_TYPE_INT32,
    {#DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR,
    #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR})
if (_GetBAPIExport = nil) then
    MsgBox.Info("Error: GetBAPIExport", "")
    return nil
end
_GetBAPIExportField = DLLProc.Make(_myDLL, "GetBAPIExportField", #DLLPROC_TYPE_INT32,
    {#DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR,
    #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR})
if (_GetBAPIExportField = nil) then
    MsgBox.Info("Error: GetBAPIExportField", "")
    return nil
end
_GetBAPITableField = DLLProc.Make(_myDLL, "GetBAPITableField", #DLLPROC_TYPE_INT32,
    {#DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR,
    #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR})
if (_GetBAPITableField = nil) then
    MsgBox.Info("Error: GetBAPITableField", "")
    return nil
end
_ClearBAPITable = DLLProc.Make(_myDLL, "ClearBAPITable", #DLLPROC_TYPE_INT32,
    {#DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR})
if (_ClearBAPITable = nil) then

```

```

    MsgBox.Info("Error: ClearBAPITable", "")
    return nil
end

_DumpBAPITableToDBase = DLLProc.Make(_myDLL,
    "DumpBAPITableToDBase", #DLLPROC_TYPE_INT32,
    {#DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR,
    #DLLPROC_TYPE_STR})
if (_DumpBAPITableToDBase = nil) then
    MsgBox.Info("Error: DumpBAPITableToDBase", "")
    return nil
end
_SAPINIT = 1

```

**Below is an example of a Visual Basic module that declares the RFC functions of the DLL.**

---

Option Explicit

'-----

'The following declarations initialize the functions of the ESRI\_R3.DLL.

Public Declare Function vbR3Logon Lib "ESRI\_R3" \_

(ByVal Client As String, ByVal User As String, ByVal PassWord As String, \_

ByVal Language As String, ByVal Destination As String, \_

ByVal HostName As String, ByVal Gateway As String, \_

ByVal Service As String, ByVal SystemNo As Long) As Long

Public Declare Function vbR3Logoff Lib "ESRI\_R3" () As Long

Public Declare Function vbInitializeRFC Lib "ESRI\_R3" \_

(ByVal FunctionName As String) As Long

Public Declare Function vbSetRFCImport Lib "ESRI\_R3" \_

(ByVal FunctionName As String, ByVal Parameter As String, ByVal Value As String) As Long

Public Declare Function vbSetRFCImportField Lib "ESRI\_R3" \_

(ByVal FunctionName As String, ByVal StructureName As String, ByVal Field As String, \_

ByVal Value As String) As Long

Public Declare Function vbAddRFCTableRow Lib "ESRI\_R3" \_

(ByVal FunctionName As String, ByVal TableName As String) As Long

Public Declare Function vbSetRFCTableRow Lib "ESRI\_R3" \_

(ByVal FunctionName As String, ByVal TableName As String, ByVal RowNumber As Long) As Long

Public Declare Function vbSetRFCTableField Lib "ESRI\_R3" \_

(ByVal FunctionName As String, ByVal TableName As String, \_

ByVal ColumnName As String, ByVal Value As String) As Long

Public Declare Function vbCallRFC Lib "ESRI\_R3" \_

(ByVal FunctionName As String) As Long

Public Declare Function vbGetRFCTableRowCount Lib "ESRI\_R3" \_

(ByVal FunctionName As String, ByVal TableName As String, ByVal ReturnCode As String) As Long

Public Declare Function vbGetRFCExport Lib "ESRI\_R3" \_

(ByVal FunctionName As String, ByVal Parameter As String, ByVal Result As String, \_

ByVal RType As String) As Long

Public Declare Function vbGetRFCExportField Lib "ESRI\_R3" \_

(ByVal FunctionName As String, ByVal StructureName As String, ByVal Field As String, \_

ByVal Result As String, ByVal RType As String) As Long

Public Declare Function vbGetRFCTableField Lib "ESRI\_R3" \_

(ByVal FunctionName As String, ByVal TableName As String, \_

ByVal ColumnName As String, ByVal Result As String, ByVal RType As String) As Long

Public Declare Function vbDumpSQLTableToDBase Lib "ESRI\_R3" \_

(ByVal FileName As String) As Long

Public Declare Function vbDumpRFCTableToDBase Lib "ESRI\_R3" \_

(ByVal FunctionName As String, ByVal TableName As String, ByVal FileName As String) As Long

Public Declare Function vbClearRFCTable Lib "ESRI\_R3" \_

(ByVal FunctionName As String, ByVal TableName As String) As Long